CS362-004

Final Project: **Part-B**
The final project is designed to check for your cumulative understanding.

Part-B: Due date **Monday, June 11th (06.11.2018)** 23:59 pm (PST).

Note:

- Use the **URLValidatorInCorrect** folder for **Part-B**.
- Submit **one copy** per group to the following locations: Canvas (pdfs) and the GitHub class repository (code).
- Add a comment in **Canvas** and give the URL of the student fork who submitted your project in the GitHub (under final project).
- Do not forget to write your group members in the pdf files!
- The code and documentation for the Apache Commons Validator is available in this link **http://commons.apache.org/proper/commons-validator/**
- The github account is available in this link https://github.com/apache/commons-validator

Preliminaries and Warm up

Get the latest files from the class repository by running

```
git remote add upstream "https://github.com/aburasa/CS362-004-S2018"
git checkout master            # make sure we're on the master branch
git fetch upstream             # pull any information about changes in upstream
git merge upstream/master -m "Sync" # merge new files
```

You should now have new files in /FinalProject/

Copy FinalProject folder into your projects/youonid/FinalProject/

The directory "/FinalProject/" contains 2 different folders **URLValidatorCorrect** and **URLValidatorInCorrect**. We will be using the **URLValidatorInCorrect** for **Part-B**.

Part-B: Testing URL Validator

You are provided a buggy version of URLValidator. You need to find out as many bugs as you can in this bad URLValidator. In the **Part-A**, I have provided the current test framework that Apache commons team had to test URLValidator. We need to assume that all those tests don't exist. Your team is a testing company and client comes to you with URLValidator implementation and asks for your help to make it bug-free. You need to just concentrate on **isValid method**, the one that is tested in testIsValid() method. *Your task is to find out **two** the bugs*, find out failure causes and provide bug reports. *You don't need to **fix** any of the bugs*. Developers will do it.

You can use any methodology that you learn during the class to test it. To stay consistent let us do it this way:

First, just do manual testing. Call the valid method of URLValidator with different possible valid/invalid inputs and see if you find a failure. (**15 points**)

Second, come up with good input partitioning. Try to provide a varying set of inputs that partition the overall input set well. Did you find any failures? You can call valid method once or more for each partition. (**15 points**)

Third, do programming based testing. Write few unit test cases. You can have some sort of loop in your unit test and test different URL with each instance of the loop. Something very similar to testIsValid() but your own logic and idea. Even a single test will be sufficient if you write it like testIsValid() method. Did you find any failures? (**20 points**)

Submit a report called **ProjectPartB.pdf** in Canvas contains the following Sections: (**40 points**)

- **Methodology Testing** (**15 points**)
  - Clearly, write about your methodology of testing (manual, partition and programming basic).
  - In the report mention the name of your test functions that implement each methodology.
  - For manual testing, provide some of your (not all) urls.
  - For partitioning, mention your partitions with reasons and some of the urls that represent each partition.

- **Bug Report** (**10 points**)
  - Write bug report for each of the bugs you found. You need to write about the following:
    1. What is the failure?
    2. How did you find it and describe the test case that detected it?
    3. What is the cause of that failure? Explain what part of the code is causing it?
    4. You can attach a picture/snapshot of the faulty code.
- **Debugging** (**10 points**)
  - When you find out any failure, debug using Eclipse/IntelliJ debugger or any other tool and try to localize its cause. Provide at what line/lines in what file the failure manifested itself. Did you use any of Agan's principle in debugging URLValidator?
  - Provide your debugging details for each bug.
- **Team Work** (**5 points**)
  - Write about how did you work in the team? How did you divide your work? How did you collaborate?
  - The contribution of each of your team members.

Submission instructions ():

- **Canvas**
  - **ProjectPartB.pdf** (due date is Monday, December 4th at 23:59 pm) (90 **Points**)

CS362-004

- **The class github repository** **(10 points)**
  - Submit your unit tests/random tests on github under **projects/your-onid/URLValidatorInCorrect/** folder.
  - Create a new **branch** of your repository called "**youronid-finalproject**" contains your final submission. This branch must be created before the due date to receive credit.

General Notes:

- ***only one submission for each group (i.e., one person needs to submit Part-a and Part-b)***
- **Write your names and onids (all group members) in the Canvas comment and the pdf files.**
- Add a comment in **Canvas** and give the URL of the student fork who submitted your project in the GitHub(under final project).

(-10 for not submitting the URL)